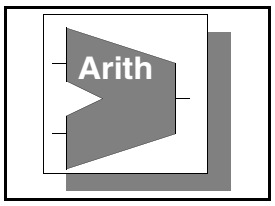
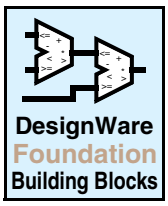




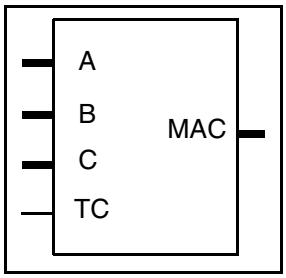
DW02_mac

Multiplier-Accumulator

Version, STAR, and myDesignWare Subscriptions: IP Directory

Features and Benefits

- Parameterized word length
- Unsigned and signed (two's-complement) data operation
- Inferable using a function call



Revision History

Description

DW02_mac is a multiplier-accumulator. It multiplies a number A by a number B, and adds the result to a number C to produce a result MAC.

The input control signal TC determines whether the inputs and outputs are interpreted as unsigned (TC = 0) or signed (TC = 1) numbers.

To extend the accuracy of the accumulator beyond A × B, use DW02_prod_sum1 Multiplier-Adder in place of DW02_mac.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
A	*A_width* bits	Input	Multiplier
B	*B_width* bits	Input	Multiplicand
C	*A_width* + *B_width* bits	Input	Addend
TC	1 bit	Input	Two's complement control ■ 0 = Unsigned ■ 1 = Signed
MAC	*A_width* + *B_width* bits	Output	MAC result (A × B + C)

Table 1-2 Parameter Description

Parameter	Values	Description
A_width	≥ 1	Word length of A
B_width	≥ 1	Word length of B

Table 1-3 Synthesis Implementations[a]

Implementation Name	Function	License Feature Required
pparch	Delay-optimized flexible Booth Wallace	DesignWare
apparch	Area-optimized flexible architecture that can be optimized for area, for speed, or for area, speed	DesignWare

a. During synthesis, Design Compiler selects the appropriate architecture for your constraints. However, you may force Design Compiler to use one of the architectures described in this table.

Table 1-4 Simulation Models

Model	Function
DW02.DW02_MAC_CFG_SIM	Design unit name for VHDL simulation
dw/dw02/src/DW02_mac_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW02_mac.v	Verilog simulation model source code

Table 1-5 Functional Description

TC	A	B	MAC
0	A (unsigned)	B (unsigned)	(A × B) + C (unsigned)
1	A (two's complement)	B two's complement)	(A × B) + C (two's complement)

Related Topics

- Math – Arithmetic Overview
- DesignWare Building Block IP User Guide

HDL Usage Through Function Inferencing - VHDL

```
library IEEE, DWARE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use DWARE.DW_foundation_arith.all;

entity DW02_mac_func is
  generic(wordlength1 : integer:=8; wordlength2: integer := 8);
  port(func_A   : in std_logic_vector(wordlength1-1 downto 0);
       func_B   : in std_logic_vector(wordlength2-1 downto 0);
       func_C   : in std_logic_vector(wordlength1+wordlength2-1 downto 0);
       func_TC  : in std_logic;
       MAC_func : out std_logic_vector(wordlength1+wordlength2-1 downto 0) );
end DW02_mac_func;

architecture func of DW02_mac_func is
begin

  process (func_A,func_B,func_C, func_TC)
  begin
    if func_TC = '1'  then
      MAC_func <= std_logic_vector(DWF_mac(SIGNED(func_A),
                                   SIGNED(func_B), SIGNED(func_C)) );
    else
      MAC_func <= std_logic_vector(DWF_mac(UNSIGNED(func_A),
                                   UNSIGNED(func_B), UNSIGNED(func_C)) );
    end if;
  end process;
end func;
```

HDL Usage Through Function Inferencing - Verilog

```
module DW02_mac_func (func_A, func_B, func_C, func_TC, MAC_func);
  parameter func_A_width = 8;
  parameter func_B_width = 8;

  // Passes the widths to the multiplier-accumulator function
  parameter A_width = func_A_width;
  parameter B_width = func_B_width;

  // Please add search_path = search_path + {synopsys_root + "/dw/sim_ver"}
  // to your .synopsys_dc.setup file (for synthesis) and add
  // +incdir+$SYNOPSYS/dw/sim_ver+ to your verilog simulator command line
  // (for simulation).
  `include "DW02_mac_function.inc"

  input [func_A_width-1 : 0]              func_A;
  input [func_B_width-1 : 0]              func_B;
  input [func_A_width+func_B_width-1 : 0] func_C;
  input                                   func_TC;

  output [func_A_width+func_B_width-1 : 0] MAC_func;

  assign MAC_func = (func_TC) ? DWF_mac_tc(func_A, func_B, func_C) :
                                DWF_mac_uns(func_A, func_B, func_C);

endmodule
```

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DWpackages.all;
use DWARE.DW_foundation_comp.all;

entity DW02_mac_inst is
  generic ( inst_A_width : NATURAL := 8;
            inst_B_width : NATURAL := 8 );
  port (inst_A  : in std_logic_vector(inst_A_width-1 downto 0);
        inst_B  : in std_logic_vector(inst_B_width-1 downto 0);
        inst_C  : in std_logic_vector(inst_A_width+inst_B_width-1 downto 0);
        inst_TC : in std_logic;
      MAC_inst: out std_logic_vector(inst_A_width+inst_B_width-1 downto 0) );
end DW02_mac_inst;

architecture inst of DW02_mac_inst is
begin

  -- Instance of DW02_mac
  U1 : DW02_mac
    generic map ( A_width => inst_A_width, B_width => inst_B_width )
    port map ( A => inst_A, B => inst_B, C => inst_C,
               TC => inst_TC, MAC => MAC_inst );
end inst;

-- pragma translate_off
configuration DW02_mac_inst_cfg_inst of DW02_mac_inst is
  for inst
  end for; -- inst
end DW02_mac_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW02_mac_inst( inst_A, inst_B, inst_C, inst_TC, MAC_inst );

  parameter A_width = 8;
  parameter B_width = 8;

  input [A_width-1 : 0] inst_A;
  input [B_width-1 : 0] inst_B;
  input [A_width+B_width-1 : 0] inst_C;
  input inst_TC;
  output [A_width+B_width-1 : 0] MAC_inst;

  // Instance of DW02_mac
  DW02_mac #(A_width, B_width)
    U1 ( .A(inst_A), .B(inst_B), .C(inst_C), .TC(inst_TC), .MAC(MAC_inst) );

endmodule
```

Revision History

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

For a version of this datasheet with visible change bars, click here.

Date	Release	Updates
March 2019	DWBB_201903.0	■ Removed Table 1-4, “Obsolete Synthesis Implementations”
January 2019	DWBB_201806.5	■ Updated example in “HDL Usage Through Component Instantiation - VHDL” on page 5 ■ Added this Revision History table and the document links on this page

Copyright Notice and Proprietary Information

© 2022 Synopsys, Inc. All rights reserved. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.

All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
www.synopsys.com